基于网格的并行FFT计算研究

陈小飞,徐宏炳
(东南大学 计算机科学与工程学院,江苏 南京 210096)

摘 要:快速傅里叶变换(FFT)在科学和工程领域有着广泛的应用。在网格环境下进行并行FFT计算可以提高运算速度,促进FFT的应用。在介绍了网格计算发展状况的基础上,详细阐述了基于网格的分布式并行计算。实验以FFT算法为背景,在Globus Toolkit 4平台下实现了并行FFT计算,并对实验数据作了分析,说明了基于网格的并行FFT计算的可行性。最后指出网格资源调度对并行计算的重要性。

关键词:网格;并行计算;Globus Toolkit 4;FFT;资源调度
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2008)03-0067-04

Research of Parallel FFT Computing Based on Grid

CHEN Xiao-fei, XU Hong-bing
(College of Computer Science & Engineering of Southeast University, Nanjing 210096, China)

Abstract: Fast Fourier Transform(FFT) is widely used in the science and engineering fields. Parallel FFT computing based on grid can improve computational speed and promote the use of FFT. Introduces the development of grid computing, then explicates distributed parallel computing based on grid. Taking FFT algorithm as backdrop, the experiment implements parallel FFT computing on the Globus Toolkit 4 platform, and analyses the experiment data, and then demonstrates the feasibility of parallel FFT computing based on grid. In the last, point out that grid resource scheduling is great importance to parallel computing.

Key words: grid; parallel computing; Globus Toolkit 4FFT; resource scheduling

0 引 言

网格(Grid)是一个集成的计算与资源环境,或者说是一个计算资源池[1],它旨在支持高性能计算和高度的资源共享。基于网格实现分布式并行计算是必然的趋势。

Globus作为国际上最具影响力的与网格计算(Grid Computing)相关的项目之一,其项目组和IBM于2002年共同提倡了一个全新的网格标准OGSA。OGSI是作为OGSA的核心规范提出来的,但近两年的实践证明,OGSI存在明显的不足[2]。为了解决OGSI和Web服务之间存在的矛盾,提出了Web服务资源框架(WSRF)。

WSRF是一组规范,提供一种标准方法,在Web服务应用程序的本质上"无状态"环境中与"有状态"资源交互。WSRF推出的目的在于,定义出一个通用且开放的架构,利用Web服务对具有状态属性的资源进行存取,并包含描述状态属性的机制,另外也包含如何将机制延伸至Web服务中的方式。随着WSRF规范的提出,Globus联盟于2005年发布了Globus Toolkit 4(GT4)工具包[3],该工具包实现了新的Web服务标准,如WSRF和Web Services Notification(WSN)。

1 网格和分布式并行计算

提供网格服务的计算节点组成强大虚拟计算资源,完成由客户端提交的并行计算任务。其中,每个节点即一个并行计算节点,提供一份执行子计算任务的服务。客户端通过GT4平台,并发地调用这些网格服务实例。

由于每个并行节点构建于GT4平台,支持不同的计算机和操作系统,方便异构环境之间的网络通信,充分地利用现有的网络资源完成分布式并行计算。传统的并行计算环境相对集中,资源相对固定,与此相比,利用网格基础设施作为分布式并行计算环境的优点在于:充分利用Internet的资源实现大型问题的分布式并行计算,屏蔽了计算资源之间的异构性[4],实现方法简单,实现代价减小。

收稿日期:2007-06-29
作者简介:陈小飞(1982-),男,江苏兴化人,硕士研究生,研究方向为网格计算、并行计算;徐宏炳,教授,主要研究领域为网格计算、数据集成、图形学等。

1.1 基于网格的并行计算分类

根据应用的需求,基于网格的分布式并行计算,可以分成三类:独立并行型、紧密耦合并行程序和松散耦合并行服务。独立并行型是最简单的应用程序,其中所有作业独立工作,每个作业产生的数据就是结果的一部分,而不需要被其他作业处理。紧密耦合并行程序是供具有特殊需要的应用程序使用的,它设计从底层开始就支持并紧密集成了并行处理,可以在构成应用程序的各种不同的程序中使用,但这种应用程序不太普遍且比较难以分解。松散耦合并行服务是将面向服务的架构和并行批处理模型结合在一起来实现并行计算,即可以通过并行服务来实现[5]。

1.2 关键技术

(1)适应性算法。网络最重要的两个属性是带宽和延迟。网格的格计算节点间是分布的,相比并行机内部以微秒为单位的通信延迟,这种域延迟让人感到头疼。虽然无 法削除这种障碍,但在一定程度上削弱问题的影响是可能的,其基本方法是分析应用和网格的特性,用一些适应性算法,减少对网络带宽的依赖,隐藏部分网络延迟。

(2)资源调度策略。网格的能力,最终由应用程序的运行性能来体现。为了获得更好运行能力,要仔细分析应用的特性、计算资源的特点,网络和输入输出等问题。然后在各因素间做出合理匹配,并在运行过程中不断优化。

(3)容错机制。基于网格的分布式计算环境由若干台计算机用 Internet 连接而成,每台计算机是自治的,与集中式系统相比更为强壮,一个单元或资源的故障不影响其他资源的正常功能。但由于 Internet 通信的不可靠性,需要提供容错措施。理论上可以通过全局的检查点实现网格的容错,但实现难度大,较为可行的方法是通过网格协议和应用程序自身来实现容错[6]。

1.3 实现分布并行计算的方法

基于对上述的综合分析,在 GT4 下进行分布式并行应用开发方法如下[7]:

(1)分解并行任务,创建多个可并行的网格服务。每个网格服务完成一个或多个并行任务。

(2)通过继承 DThread 类或实现 Runnable 接口来创建一个分布式线程。

(3)分布线程的 run()方法中创建一个网格服务实例,即部署并行计算代码到网格节点中。

(4)创建网格客户应用,负责管理启动多个分布线程。

(5)多个并行网格服务执行并行任务。网格服务之间以通知机制来完成并行任务之间消息和数据的通信。

(6)网格服务节点返回计算结果给网格客户应用。

2 并行 FFT 算法描述

并行 FFT 算法,主要包括并行 FFT 迭代算法和并行 FFT 递归算法[8]。

2.1 并行 FFT 迭代算法

假设有 n 个数构成的数列,p 个处理节点,通常情况下,n 和 p 应该是 2 的幂次方。

每个处理节点包含 m 个连续数据块($m = n/p$)。

输入:$a = (a_0, a_1, \cdots, a_{n-1})$

输出:$b = (b_0, b_1, \cdots, b_{n-1})$

```
Begin
/* 对所有节点 id(id=0,1,…,p-1)执行如下的算法 */
j=0
/* logn-1~logm 步不需要通信,m=n/p,所以 logm=logn-logp
logn-1~logm ->logm+logp-1~logm ->logp~10 */
(1)for h=logp-1 downto 0 do
  (1.1)t=2^h,j=j+1,v=2^j,l=n/v,q=n/l=v,z=w^q/2
  (1.2)if((id mod t)=(id mod 2t)) then
    1.rank=id + p/v
    2.接收由 rank 号节点发来的数据块,并将它们存放到 c[id*m+l]到 c[id*m+l+m-1]中
    3.for k=id*m to id*m + m-1 do
    c[k]=c[k]+c[k+l]
    c[k+l]=(c[k]-c[k+l])*z(k mod l)
    end for
    4.将存于 c[id*m + l]到 c[id*m+l+m-1]中的数据块发送到 rank 号节点
    else
    5.将数据块发送到(id-p/v)号节点
    6.接收由(id-p/v)号节点发来的数据块
    end if
  end for
(2)for i=logm-1 downto 0 do
  (2.1) l=2^i,q=n/l,z=w^q/2
  (2.2) for k=id*m to id*m+m-1 do
    if((k mod l)=(k mod 2l)) then
      c[k]=c[k]+c[k+l]
      c[k+l]=(c[k]-c[k+l])*z(k mod l)
    end if
  end for
end for
(3)/*汇总各节点的数据*/
  (3.1)其他节点将数据块发送给 id=0 号节点
  (3.2) for k=0 to n-1 do
```

```
        b[r(k)] = c[k] /* r(k)为 k 的位反 */
      end for
End
```

2.2 并行 FFT 递归算法

假设有 n 个数构成的数列,p 个处理节点,每个处理节点包含一个连续的数据块。

输入:$a = (a_0, a_1, \cdots, a_{n-1})$

输出:$b = (b_0, b_1, \cdots, b_{n-1})$

```
FFT (a,beginPos,endPos,ω,b,leftPos,rightPos,n)
Begin
  (1)if beginPos = endPos then
      for i = leftPos to rightPos do
        b[i] = a[beginPos]
      end for
  (2)else if beginPos + 1 = endPos then
      for i = leftPos to rightPos do
        b[i] = a[beginPos] + a[endPos] * ω[i]
      end for
  (3) else
    (3.1) midPos = (beginPos + eodPos)/2
    (3.2)对数组 a 中从 beginPos 到 endPos 位置的数据进行处理,偶序列数移置前半部,奇序列数移置后半部。
    (3.3) for i = leftPos to rightPos do
        X[i] = ω[i] * ω[i]
      end for
    (3.4) FFT (a,0,midPos,X,Y1,leftPos,rightPos,n)
    (3.5) FFT (a,midPos+1, n-1,X,Y2,leftPos,rightPos,n)
    (3.6) for i = leftPos to rightPos do
        b[i] = Y1[i] + Y2[i] * ω[i]
      end for
  end if
End
```

3 实验数据分析

在 4 台异构的网络节点组成的网格环境中,以 $2^n(n = 6,\cdots,19)$ 的数据量分别为输入,进行并行 FFT 计算实验,实验数据见表 1 和表 2,网格节点情况见表 3。

依据实验数据绘制线性图 1 和线性图 2。

根据表 1 和图 1 可知,FFT 的迭代算法在效率上明显优于递归算法,这是因为递归算法通常需要执行大量的过程调用,并在堆栈中保存所有返回过程的中间变量,效率往往较低,并且两节点的递归算法虽然比单节点递归算法效率有提高,但相比迭代算法效率依然很低,所以递归算法不适用于分布式并行计算。

根据表 2 和图 2,对于 FFT 的迭代算法,当数据量达到 $2^{14} = 16384$ 时,两个节点的计算效率高于单个节点;当数据量达到 $2^{18} = 262144$ 时,四个节点的运行效率高于两个节点的运行效率。即随着问题规模的增大,在 GT4 环境下,并行计算的效率要高于串行计算。但并非节点数越多效率越高,因为节点间通信也需要时间,应该根据问题规模的大小来选择节点个数。

表 1　迭代和递归算法实验数据

数据量	迭代算法(ms)		递归算法(ms)	
	单节点	两节点	单节点	两节点
2^6	0	110	0	313
2^7	4	121	16	391
2^8	15	125	31	390
2^9	18	126	109	375
2^{10}	31	139	750	796
2^{11}	39	165	3797	2312
2^{12}	55	178	17016	7591
2^{13}	112	179	65578	30875
2^{14}	289	230		
2^{15}	598	334		
2^{16}	1151	621		
2^{17}	2369	130		
2^{18}	4832	3425		
2^{19}	13685	7978		

表 2　不同节点迭代算法实验数据

数据量	单节点迭代算法(ms)	两节点迭代算法(ms)	四节点迭代算法(ms)
2^6	0	110	172
2^7	4	121	187
2^8	15	125	188
2^9	18	126	219
2^{10}	31	139	234
2^{11}	39	165	265
2^{12}	55	178	281
2^{13}	112	179	281
2^{14}	289	230	297
2^{15}	598	334	449
2^{16}	1151	621	713
2^{17}	2369	1309	1541
2^{18}	4832	3425	2816
2^{19}	13685	7978	5781

表 3　系统配置情况

节点	CPU	RAM	OS	Grid
1	Intel P4 2.8G	512M DDR	RH Linux 9.0	GT4
2	Intel P4 2.6G	512M DDR	RH Linux 9.0	GT4
3	Intel P4 3.0G	512M DDR	RH Linux 9.0	GT4
4	Intel P4 2.8G	512M DDR	RH Linux 9.0	GT4

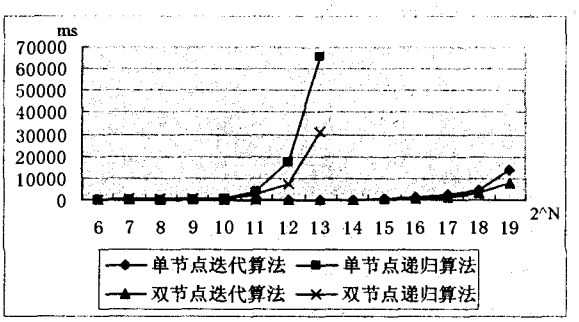

图 1　迭代算法和递归算法比较

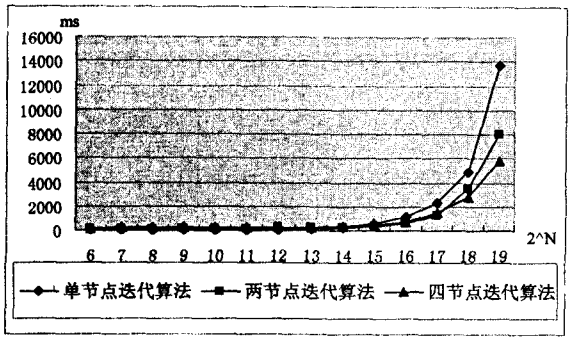

图 2 不同节点迭代算法比较

同时,以上也表明了,在基于 GT4 的分布式并行计算环境下,使用合适的并行算法进行分布式并行计算是可行的。

4 结束语

文中 GT4 环境下,实现了并行 FFT 迭代算法和并行 FFT 递归算法,并对实验数据进行了分析,说明了网格平台下进行分布式并行计算的可行性。但是如何以更小的代价来实现高性能的分布式并行计算? 网格环境中资源调度[9]对计算性能起着至关重要的作用,高效的调度算法或策略可以充分利用网格系统的处理能力,从而提高应用程序的性能。这正是下一步要研究的课题。

参考文献:

[1] Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure[DB/OL]. San Francisco, CA: Morgan Kaufmann, 1999. http://www.gridforum.org.

[2] 易 明,金 海. 基于 WSRF 的 Web 服务资源的设计[J]. 计算机工程,2006,32(23):262 - 264.

[3] Foster I. Globus Toolkit Version 4: Software for Service - Oriented Systems[J]. Journal of Computer Science and Technology, 2006, 21(4): 513 - 520.

[4] Foster I, Kesselman C. The Physiology of the Grid[M]//Grid Computing: Making the Global Infrastructure a Reality. Chinchester, UK: John Wiley & Sons, 2003: 217 - 249.

[5] 林伟伟,张志立,齐德昱. 基于网格的分布并行计算策略[J]. 计算机工程,2006,32(9):104 - 106.

[6] 都志辉,陈 渝,刘 鹏. 网格计算[M]. 北京:清华大学出版社,2002.

[7] 林伟伟,齐德昱,王振宇. 网格计算环境下分布并行计算的一种实现方法[J]. 计算机工程与应用,2005,41(27):32 - 34.

[8] 陈国良. 并行算法实践[M]. 北京:高等教育出版社,2003.

[9] 赵宏伟,邵一川. 基于 Globus 网格服务的作业调度的实现[J]. 计算机工程与设计,2006,27(9):1665 - 1668.

(上接第 66 页)

证明文中的保持共享对象一致性的方法较好地满足了一致性模型提出的收敛性、因果顺序性和操作目的性的要求。因为本地的操作可以立即执行,所以很好地满足了实时协同设计的响应性要求。通过消息驱动机制进行协同,网络传输的数据主要是信息量很少的标准消息,所以网络负载轻,同时各协同站点在本地调用标准消息映射的实体函数执行,提高了响应性。因为系统只保留最高角色权限的操作效果,也就不再有非几何属性操作冲突后的问题,而且每次需要撤销用户的操作效果时,只需要 UNDO 一次即可。

参考文献:

[1] Preston J A, Prasad S K. Exploring Communication Overheads and Locking Policies in a Peer - to - Peer Synchronous Collaborative Editing System[C]//ACM Southeast Conference. New York, NY, USA: ACM Press, 2005: 286 - 394.

[2] Ellis C A, Gibbs S J. Concurrency control in groupware systems[C]//ACM SIGMOD Conference on Management of Data. Seattle, WA: ACM, 1989: 399 - 407.

[3] Sun C, Ellis C A. Operational transformation in real - time group editors: Issues, algorithms, and achievements[C]//In: Poltrock S, Grudin J, eds. Proceedings of ACM 1998 Conference on Computer Supported Cooperative Work. New York: ACM Press, 1998: 59 - 68.

[4] Enokido T, Takizawa M. Concurrency Control Based on Significancy on Roles[C]//The IEEE 11th International Conference on Parallel and Distributed systems. New York: ACM Press, 2005: 196 - 202.

[5] Sabbir A, Ravindran K N. Concurrency Control Frameworks for Interactive Sharing of Data Spaces in Real - time Distributed Collaborations[R]. USA: Department of Computer Science, City University of New York, 2005: 16 - 26.

[6] Sun C, Chen D. Consistency Maintenance in Real - time Collaborative Graphics Editing Systems[J]. ACM Transactions on Computer - Human Interaction, 2002, 9(1): 1 - 10.

[7] Dou Wanfeng, Zhu Ming. Cooperative multi - versioning technique based on version replication[C]//Proc. of the 10th International Conf. on Computer Supported Cooperative work in Design. Nanjing, China: [s.n.], 2006: 159 - 164.

[8] Lamport L. Time, clocks, and the ordering of events in a distributed system[J]. ACM, 1978, 21(7): 558 - 565.

[9] 余春艳,侯宏仑,吴明晖,等. 协同设计中以实体为中心的并发操作控制机制[J]. 计算机辅助设计与图形学学报,2004,16(9):1289 - 1294.